# Exercise 3: Sorting Customer Orders

# 1.UNDERSTAND THE PROBLEM:

**Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).**

**Bubble Sort:**

* **Description:** Repeatedly compares adjacent elements and swaps them if they are in the wrong order. This process is repeated until the array is sorted.
* **Time Complexity:**
  + **Best Case:** O(n) – when the array is already sorted.
  + **Average Case:** O(n^2) – for typical scenarios.
  + **Worst Case:** O(n^2) – when the array is sorted in reverse order.

**Insertion Sort:**

* **Description:** Builds the final sorted array one item at a time. It picks the next item and inserts it into the correct position in the already sorted portion of the array.
* **Time Complexity:**
  + **Best Case:** O(n) – when the array is already sorted.
  + **Average Case:** O(n^2) – for typical scenarios.
  + **Worst Case:** O(n^2) – when the array is sorted in reverse order.

**Quick Sort:**

* **Description:** Uses a divide-and-conquer approach by selecting a ‘pivot’ element and partitioning the array into two sub-arrays according to whether elements are less than or greater than the pivot. The sub-arrays are then sorted recursively.
* **Time Complexity:**
  + **Best Case:** O(n log n) – when the pivot divides the array into equal halves.
  + **Average Case:** O(n log n) – for typical scenarios.
  + **Worst Case:** O(n^2) – when the pivot is the smallest or largest element repeatedly.

**Merge Sort:**

* **Description:** Divides the array into halves, recursively sorts each half, and then merges the sorted halves to produce the final sorted array.
* **Time Complexity:**
  + **Best Case:** O(n log n) – for typical scenarios.
  + **Average Case:** O(n log n) – for typical scenarios.
  + **Worst Case:** O(n log n) – for typical scenarios.

# 2. ANALYSIS:

**Compare the performance (time complexity) of Bubble Sort and Quick Sort.**

 **Bubble Sort:**

* **Best Case:** O(n) – when the array is already sorted (optimizable with a flag).
* **Average Case:** O(n^2) – due to nested loops.
* **Worst Case:** O(n^2) – when the array is sorted in reverse.

 **Quick Sort:**

* **Best Case:** O(n log n) – when the pivot divides the array into equal halves.
* **Average Case:** O(n log n) – for typical scenarios.
* **Worst Case:** O(n^2) – when the pivot is the smallest or largest element repeatedly.

**Discuss why Quick Sort is generally preferred over Bubble Sort.**

* **Efficiency:** Quick Sort has a significantly better average and best-case time complexity (O(n log n)) compared to Bubble Sort's O(n^2).
* **Scalability:** Quick Sort is more efficient for large datasets and can handle them better due to its divide-and-conquer approach.
* **Adaptability:** Quick Sort can be optimized with different pivot selection strategies (e.g., median-of-three), whereas Bubble Sort does not have such optimizations.

In general, Quick Sort is preferred for its superior performance on larger datasets